

 **GETTING STARTED GUIDE**

Altair Embed[®] Personal 2025.2

Blinking the built-in LED on an Arduino

Intellectual Property Rights Notice:

Copyright ©1986-2025 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of the intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions.

In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. or its affiliates. Additionally, all non-Altair marks are the property of their respective owners. If you have any questions regarding trademarks or registrations, please contact marketing and legal.

This Intellectual Property Rights Notice does not give you any right to any product, such as software, or underlying intellectual property rights of Altair Engineering Inc. or its affiliates. Usage, for example, of software of Altair Engineering Inc. or its affiliates is governed by and dependent on a valid license agreement.

Altair® HyperWorks®, a Design & Simulation Platform

Altair® AcuSolve® ©1997-2025
Altair® Activate® ©1989-2025
Altair® Automated Reporting Director™ ©2008-2022
Altair® Battery Damage Identifier™ ©2019-2025
Altair® CFD™ ©1990-2025
Altair® Compose® ©2007-2025
Altair® ConnectMe™ ©2014-2025
Altair® DesignAI™ ©2022-2025
Altair® DSim® ©2024-2025
Altair® DSim® Cloud ©2024-2025
Altair® DSim® Cloud CLI ©2024-2025
Altair® DSim® Studio ©2024-2025
Altair® EDEM™ ©2005-2025
Altair® EEvision™ ©2018-2025
Altair® ElectroFlo™ ©1992-2025
Altair® Embed® ©1989-2025
Altair® Embed® SE ©1989-2025
Altair® Embed®/Digital Power Designer ©2012-2025
Altair® Embed®/eDrives ©2012-2025
Altair® Embed® Viewer ©1996-2025
Altair® e-Motor Director™ ©2019-2025
Altair® ESAComp® ©1992-2025
Altair® expertAI™ ©2020-2025
Altair® Feko® ©1999-2025
Altair® FlightStream® ©2017-2025
Altair® Flow Simulator™ ©2016-2025
Altair® Flux® ©1983-2025
Altair® FluxMotor® ©2017-2025
Altair® GateVision PRO™ ©2002-2025
Altair® Geomechanics Director™ ©2011-2022
Altair® HyperCrash® ©2001-2023
Altair® HyperGraph® ©1995-2025
Altair® HyperLife® ©1990-2025
Altair® HyperMesh® ©1990-2025
Altair® HyperMesh® CFD ©1990-2025
Altair® HyperMesh® NVH ©1990-2025
Altair® HyperSpice™ ©2017-2025
Altair® HyperStudy® ©1999-2025
Altair® HyperView® ©1999-2025
Altair® HyperView Player® ©2022-2025
Altair® HyperWorks® ©1990-2025
Altair® HyperWorks® Design Explorer ©1990-2025

Altair® HyperXtrude® ©1999-2025
Altair® Impact Simulation Director™ ©2010-2022
Altair® Inspire™ ©2009-2025
Altair® Inspire™ Cast ©2011-2025
Altair® Inspire™ Extrude Metal ©1996-2025
Altair® Inspire™ Extrude Polymer ©1996-2025
Altair® Inspire™ Form ©1998-2025
Altair® Inspire™ Mold ©2009-2025
Altair® Inspire™ PolyFoam ©2009-2025
Altair® Inspire™ Print3D ©2021-2025
Altair® Inspire™ Render ©1993-2025
Altair® Inspire™ Studio ©1993-2025
Altair® Material Data Center™ ©2019-2025
Altair® Material Modeler™ ©2019-2025
Altair® Model Mesher Director™ ©2010-2025
Altair® MotionSolve® ©2002-2025
Altair® MotionView® ©1993-2025
Altair® Multi-Disciplinary Optimization Director™ ©2012-2025
Altair® Multiscale Designer® ©2011-2025
Altair® newFASANT™ ©2010-2020
Altair® nanoFluidX® ©2013-2025
Altair® NLView™ ©2018-2025
Altair® NVH Director™ ©2010-2025
Altair® NVH Full Vehicle™ ©2022-2025
Altair® NVH Standard™ ©2022-2025
Altair® OmniV™ ©2015-2025
Altair® OptiStruct® ©1996-2025
Altair® PhysicsAI™ ©2021-2025
Altair® PollEx™ ©2003-2025
Altair® PollEx™ for ECAD ©2003-2025
Altair® PSIM™ ©1994-2025
Altair® Pulse™ ©2020-2025
Altair® Radioss® ©1986-2025
Altair® romAI™ ©2022-2025
Altair® RTLvision PRO™ ©2002-2025
Altair® S-CALC™ ©1995-2025
Altair® S-CONCRETE™ ©1995-2025
Altair® S-FRAME® ©1995-2025
Altair® S-FOUNDATION™ ©1995-2025
Altair® S-LINE™ ©1995-2025
Altair® S-PAD™ © 1995-2025
Altair® S-STEEL™ ©1995-2025
Altair® S-TIMBER™ ©1995-2025
Altair® S-VIEW™ ©1995-2025
Altair® SEAM® ©1985-2025
Altair® shapeAI™ ©2021-2025
Altair® signalAI™ ©2020-2025
Altair® Silicon Debug Tools™ ©2018-2025
Altair® SimLab® ©2004-2025
Altair® SimLab® ST ©2019-2025
Altair® SimSolid® ©2015-2025
Altair® SpiceVision PRO™ ©2002-2025
Altair® Squeak and Rattle Director™ ©2012-2025
Altair® StarVision PRO™ ©2002-2025
Altair® Structural Office™ ©2022-2025
Altair® Sulis™ ©2018-2025
Altair® Twin Activate® ©1989-2025
Altair® UDE™ ©2015-2025
Altair® ultraFluidX® ©2010-2025
Altair® Virtual Gauge Director™ ©2012-2025
Altair® Virtual Wind Tunnel™ ©2012-2025

Altair® Weight Analytics™ ©2013-2022
Altair® Weld Certification Director™ ©2014-2025
Altair® WinProp™ ©2000-2025
Altair® WRAP™ ©1998-2025

Altair® HPCWorks®, a HPC & Cloud Platform

Altair® Allocator™ ©1995-2025
Altair® Access™ ©2008-2025
Altair® Accelerator™ ©1995-2025
Altair® Accelerator™ Plus ©1995-2025
Altair® Breeze™ ©2022-2025
Altair® Cassini™ ©2015-2025
Altair® Control™ ©2008-2025
Altair® Desktop Software Usage Analytics™ (DSUA) ©2022-2025
Altair® FlowTracer™ ©1995-2025
Altair® Grid Engine® ©2001, 2011-2025
Altair® InsightPro™ ©2023-2025
Altair® InsightPro™ for License Analytics ©2023-2025
Altair® Hero™ ©1995-2025
Altair® Liquid Scheduling™ ©2023-2025
Altair® Mistral™ ©2022-2025
Altair® Monitor™ ©1995-2025
Altair® NavOps® ©2022-2025
Altair® PBS Professional® ©1994-2025
Altair® PBS Works™ ©2022-2025
Altair® Simulation Cloud Suite (SCS) ©2024-2025
Altair® Software Asset Optimization (SAO) ©2007-2025
Altair® Unlimited™ ©2022-2025
Altair® Unlimited Data Analytics Appliance™ ©2022-2025
Altair® Unlimited Virtual Appliance™ ©2022-2025

Altair® RapidMiner®, a Data Analytics & AI Platform

Altair® AI Hub ©2023-2025
Altair® AI Edge™ ©2023-2025
Altair® AI Cloud ©2022-2025
Altair® AI Studio ©2023-2025
Altair® Analytics Workbench™ ©2002-2025
Altair® Graph Lakehouse™ ©2013-2025
Altair® Graph Studio™ ©2007-2025
Altair® Knowledge Hub™ ©2017-2025
Altair® Knowledge Studio® ©1994-2025
Altair® Knowledge Studio® for Apache Spark ©1994-2025
Altair® Knowledge Seeker™ ©1994-2025
Altair® IoT Studio™ ©2002-2025
Altair® Monarch® ©1996-2025
Altair® Monarch® Classic ©1996-2025
Altair® Monarch® Complete™ ©1996-2025
Altair® Monarch® Data Prep Studio ©2015-2025
Altair® Monarch Server™ ©1996-2025
Altair® Panopticon™ ©2004-2025
Altair® Panopticon™ BI ©2011-2025
Altair® SLC™ ©2002-2025
Altair® SLC Hub™ ©2002-2025
Altair® SmartWorks™ ©2002-2025
Altair® RapidMiner® ©2001-2025

Altair One® ©1994-2025
Altair® CoPilot™ ©2023-2025

Altair® Drive™ ©2023-2025
Altair® License Utility™ ©2010-2025
Altair® TheaRender® ©2010-2025
OpenMatrix™ ©2007-2025
OpenPBS® ©1994-2025
OpenRadioss™ ©1986-2025

Contents

| | |
|--|----|
| What is Altair Embed® Personal | 2 |
| Supported hardware | 2 |
| The Altair Embed product family | 3 |
| Getting started with Embed Personal..... | 4 |
| Embed Personal window | 4 |
| Standard and target support blocks | 5 |
| Building a simple diagram: Blink the built-in LED on the Arduino Uno | 5 |
| Configuring the diagram for the Arduino Uno | 6 |
| Inserting blocks | 8 |
| Setting block parameters | 9 |
| Connecting blocks | 9 |
| Setting simulation parameters | 10 |
| Confirming the signal frequency..... | 11 |
| Saving your work..... | 12 |
| Compiling and linking your code..... | 13 |
| Downloading and executing the code on the Arduino Uno | 15 |
| Changing the blink frequency | 15 |
| Helpful resources | 16 |
| Videos | 16 |
| Sample diagrams..... | 16 |
| Where to go from here | 16 |

What is Altair Embed® Personal

Altair Embed Personal is a special edition of Altair Embed that is available for noncommercial use to anyone interested in a low-cost solution for programming an extensive range of microprocessors from Arduino®, AMD64, Raspberry Pi™, STMicroelectronics®, and Texas Instruments™. Whether you're preparing for a proof-of-concept project or developing small applications, Embed Personal provides all the tools and expertise you will need to succeed. With Embed Personal, you can easily read, write, and analyze data on your microprocessor, as well as run your algorithms on the hardware and interactively communicate with it as the code is executing.

Embed Personal offers all the features and capabilities found in Embed Pro; however, it is for personal or academic use only and comes with a 180-day renewable license. To see what's included, click [here](#).

Supported hardware

- Arduino Leonardo, Mega , Nano, and Uno
- AMD64
- Raspberry Pi Zero, Zero W, 1A+, 1B+, 2B, 3A+, 3B, 3B+, and 4B
- STMicroelectronics STM32® F0x, F103x, F3x, F4x, F7x, G0x, G4x, H7x, L4x, and WBx series
- Texas Instruments ARM Cortex® M3, C2000™, Delfino™, MSP430™, and Piccolo™

The Altair Embed product family

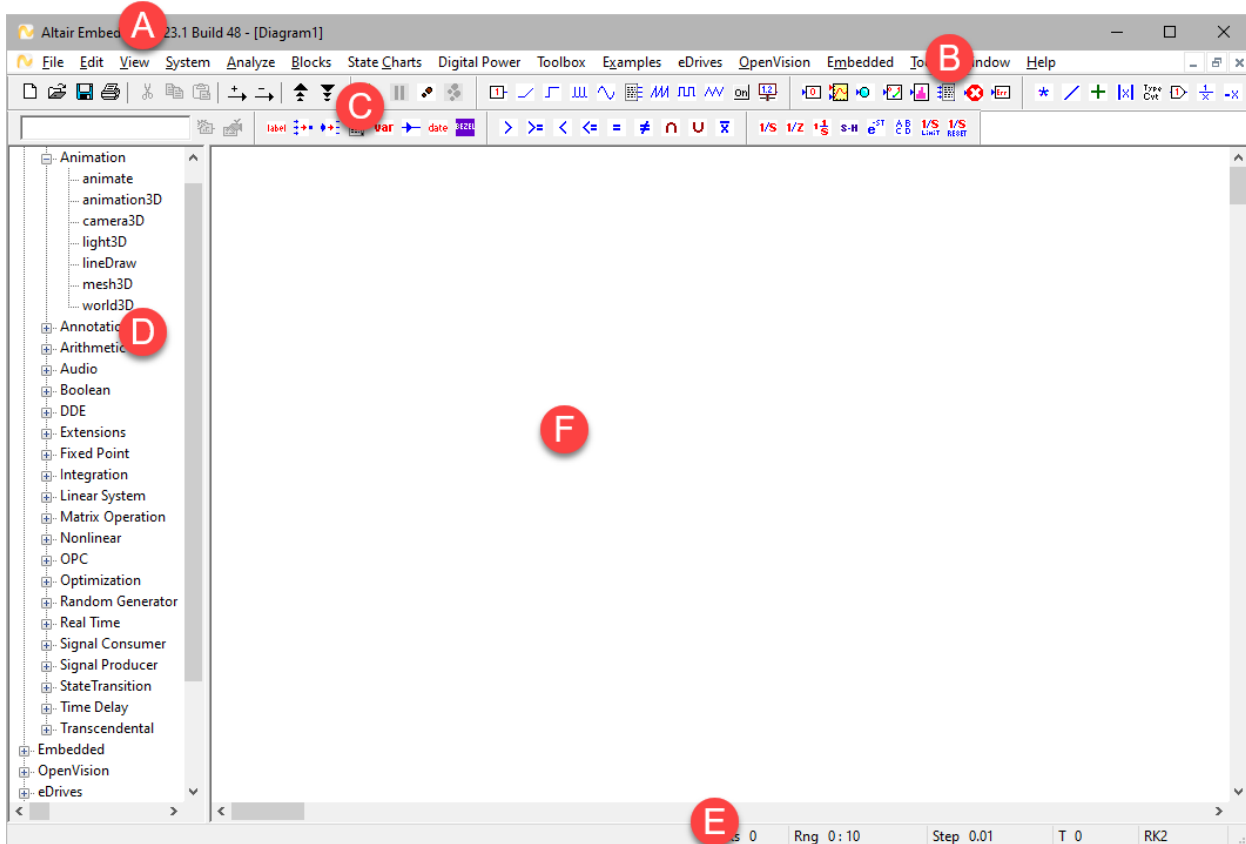
The Altair Embed product family includes the Personal, Professional (Pro), and Simulation-Only (SE) editions.

| Feature | Embed Personal | Embed Professional (Pro) | Embed Simulation-Only (SE) |
|--|----------------|--------------------------|----------------------------|
| 180-day renewable license | ✓ | | |
| Noncommercial use | ✓ | | |
| Drag-and-drop block diagram construction with 120+ standard blocks | ✓ | ✓ | ✓ |
| Continuous, discrete, and hybrid simulations | ✓ | ✓ | ✓ |
| Conditional execution of subsystems | ✓ | ✓ | ✓ |
| Interactive, batch, auto-restart, and single-step execution modes | ✓ | ✓ | ✓ |
| Fixed, adaptive, and stiff integration algorithms | ✓ | ✓ | ✓ |
| Visualization with interactive plots and stripCharts, 3D animation, 3D plots, and polar plots | ✓ | ✓ | ✓ |
| Analysis and linearization with Bode, root locus, and frequency domain plots | ✓ | ✓ | ✓ |
| Image processing and pattern recognition | ✓ | ✓ | ✓ |
| State Charts block set | ✓ | ✓ | ✓ |
| Fixed Point block set | ✓ | ✓ | |
| Digital Motor Control block set | ✓ | ✓ | |
| Data I/O with National Instruments and Measurement Computing boards, OPC server, RS 232 devices, UDP, and PCAN USB devices | ✓ | ✓ | ✓ |
| Code generation | ✓ | ✓ | |
| Arduino, AMD64, Raspberry Pi, STMicroelectronics, and Texas Instruments target support | ✓ | ✓ | |
| Generic MCU target support | ✓ | ✓ | |
| JTAG communication link | ✓ | ✓ | |
| Serial communication link | ✓ | ✓ | |

Getting started with Embed Personal


Embed Personal window

Start Embed Personal from your desktop by clicking on the Embed Personal icon.



A: The **Title bar** includes the name of the currently open diagram. As you develop diagrams with hierarchy and drill into the hierarchy, you can look at the title bar to see where you are in the diagram.

B: The **Menu bar** contains drop down menus for all commands and blocks. You can customize the menu bar with your own menu items, commands, and blocks using the Edit > Preferences > Menu Directories command.

Commands and blocks can also be accessed from the menu browser, toolbar, pop-up menus, and short-cut keyboard commands. For example, to open a new diagram, you select File > New from the menu, or you can click the  toolbar button.

C: Customizable **Toolbar buttons** are shortcuts for commonly used editing, simulation, and debugging functions.

D: The **Block and Diagram browser** provides a convenient way to insert blocks into a diagram and navigate within a multilayer diagram.

E: The **Status bar** keeps track of the number of blocks in your diagram, along with key simulation information including the simulation range, step size, elapsed simulation time, and currently selected integration algorithm.

F: The **Work area** is where you build your diagrams.

Standard and target support blocks

In Embed Personal, blocks are your basic design component. You can choose from over 200 standard mathematical, engineering, and scientific blocks, allowing you to realize system models of any degree of complexity. These blocks are categorized by function and listed in the **Blocks** menu.

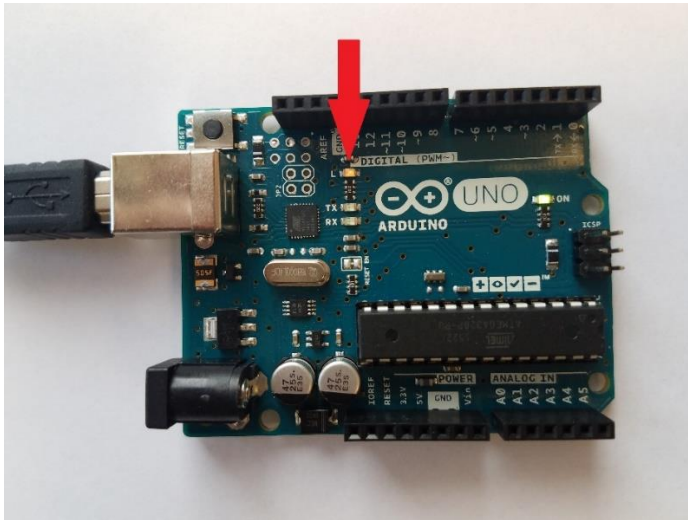
Embed Personal also provides an extensive set of target support blocks useful for controlling and reading hardware pins, allowing you to obtain analog and digital data; manipulate devices with digital and PWM outputs; and interact with devices via I2C, SPI, and UART. These blocks are listed under the **Embedded > target** menu.

There are additional block and toolbox libraries that expand Embed's modeling capabilities for mechanical and electrical systems, digital power, electric motor drives, image processing, and discrete event systems. These blocks are listed under the **Digital Power**, **eDrives**, **OpenVision**, and **State Charts** menus.

Building a simple diagram: Blink the built-in LED on the Arduino Uno

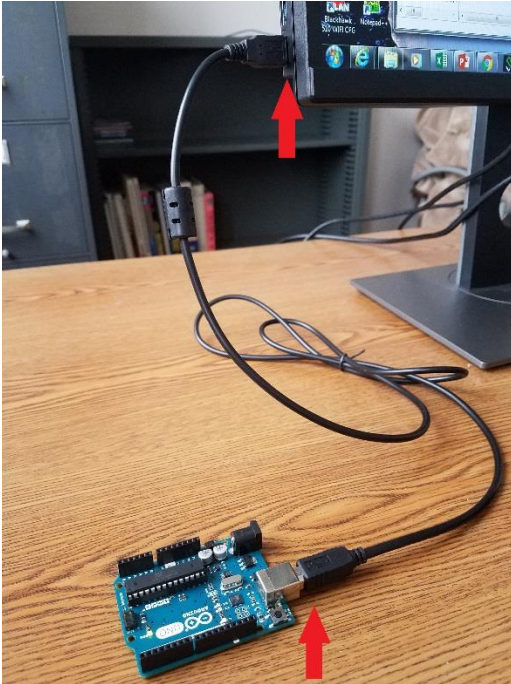
Like the *Hello, World!* program, blinking the Arduino Uno's built-in LED is used to introduce basic programming concepts. To blink the LED, you can follow the step-by-step directions below or [watch a similar online video](#).

1. Locate the built-in LED on your Arduino Uno board.



The built-in LED is connected to [port B, channel 5](#), which in turn is connected to digital pin 13 on the Arduino Uno.

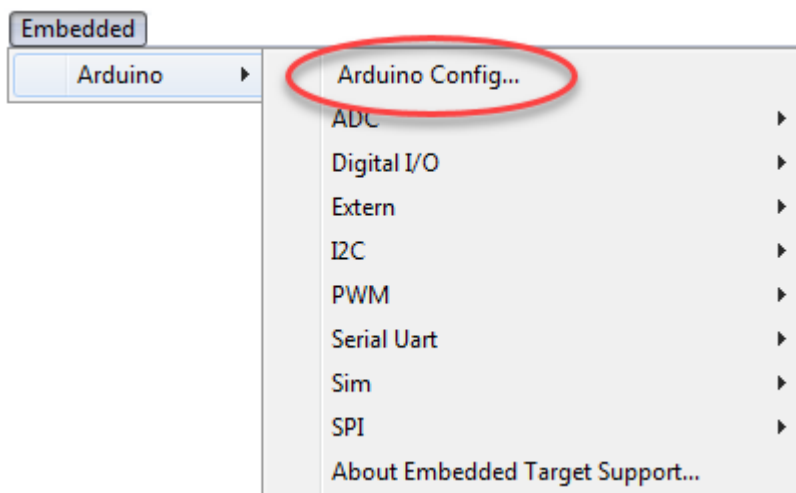
2. Attach the Arduino Uno board to your computer using a USB cable.



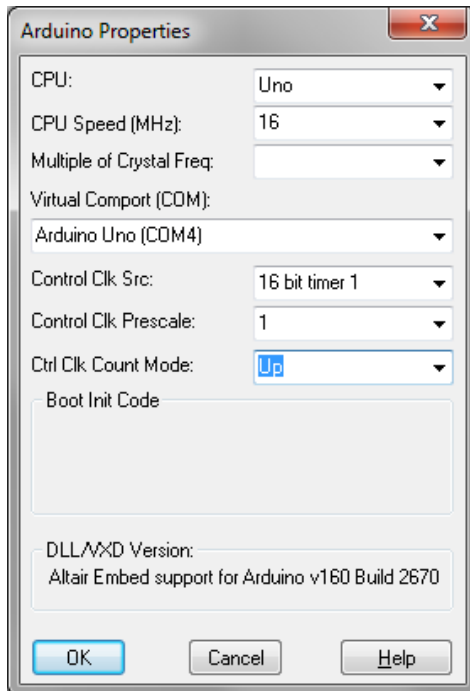
Configuring the diagram for the Arduino Uno

To construct the blink LED diagram, you use an Arduino Config block to set up the diagram for the Arduino.

1. Start Embed Personal.
A new empty diagram named *Diagram1* is created.
2. Choose **Embedded > Arduino** and click **Arduino Config**.

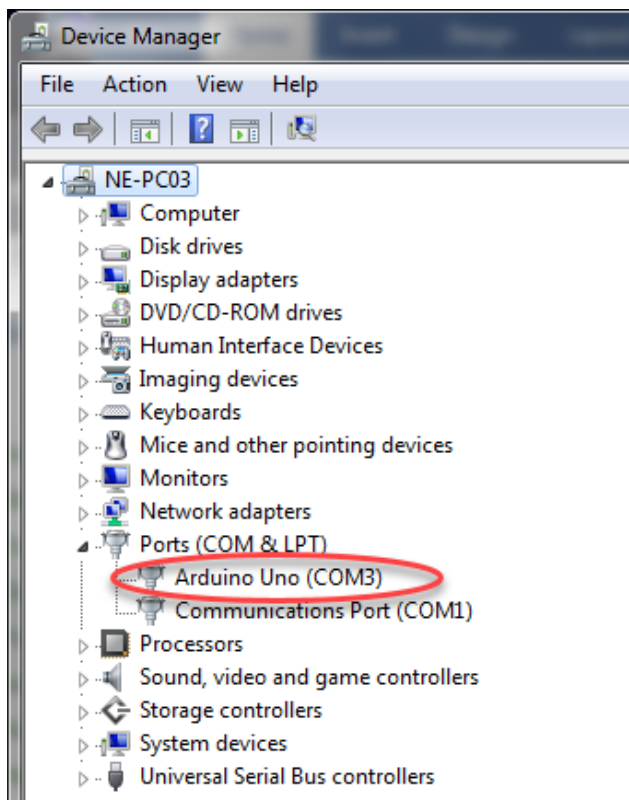


The Arduino Properties dialog appears.



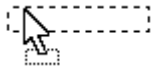
3. Under **Virtual Comport**, select the serial port number for your Arduino.

Note: If you don't know the number, click **Start > Control Panel > Device Manager** and then scroll down and click **Ports** to find it.



4. The remaining parameters in the Arduino Properties dialog are already correctly set; just click **OK**.

The pointer appears with a marquee attached to it.



5. Move the pointer to the work area and click to insert the **Arduino Config** block.

Arduino Config: Uno@16MHz

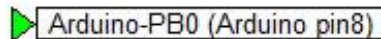
Inserting blocks

To generate a signal that goes from 0 to 1 and then back again and connect it to the Arduino Uno, you use a squareWave block and a Digital Output for Arduino block.

1. Under **Blocks > Signal Producer**, click **squareWave**.
2. Point to where you want to insert the block in the work area and **click**.
3. Under **Embedded > Arduino > Digital I/O**, click **Digital Output for Arduino**.
4. Point to where you want to insert the block in the work area and **click**.

Your diagram will look like this:

Arduino Config: Uno@16MHz

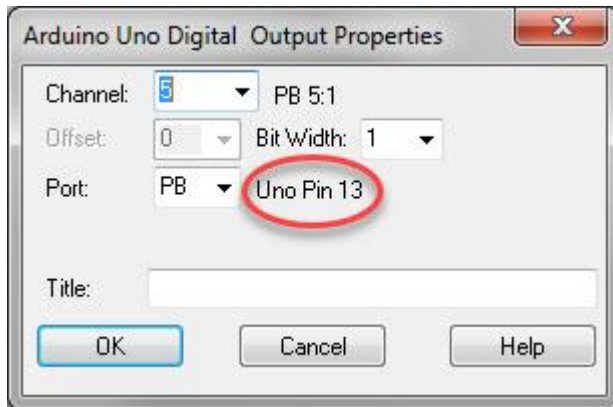


Setting block parameters

The built-in LED is connected to digital pin 13 on the Arduino Uno, which is connected to Port B, channel 5, as shown in the [Arduino Uno pin mapping schematic](#). To connect to digital pin 13:

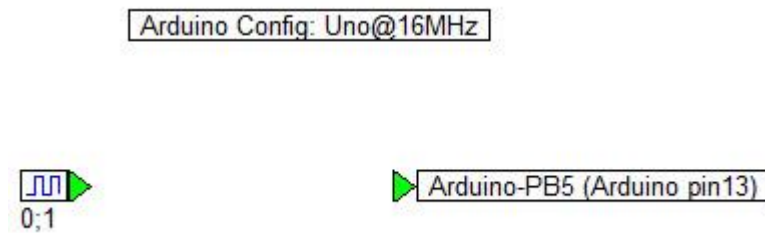
1. Right-click the **Digital Output for Arduino** block.
2. In the Arduino Output Channel dialog, do the following:
 - Under **Port**, select **PB**.
 - Under **Channel**, select **5**.

The dialog updates to show the proper connection has been made.



3. Click **OK**.

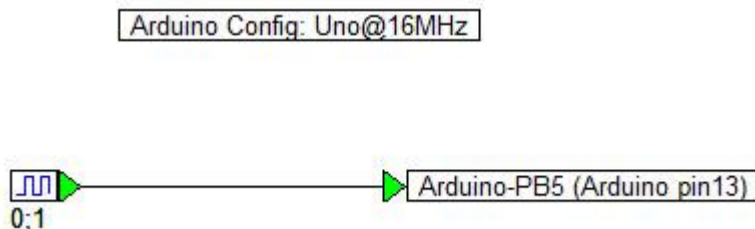
Your diagram will look like this:



Connecting blocks

By connecting blocks, you can pass signal values, or data, from one block to another. You connect blocks by creating a wire between the input and output tabs on the blocks.

1. Point to the output tab on the **squareWave** block. The pointer turns into an upward pointing arrow when it is over the tab.
2. Drag the pointer to the **input tab** on the **Digital Output for Arduino** block. When you release the mouse button, the connection is completed and your diagram will look like this:



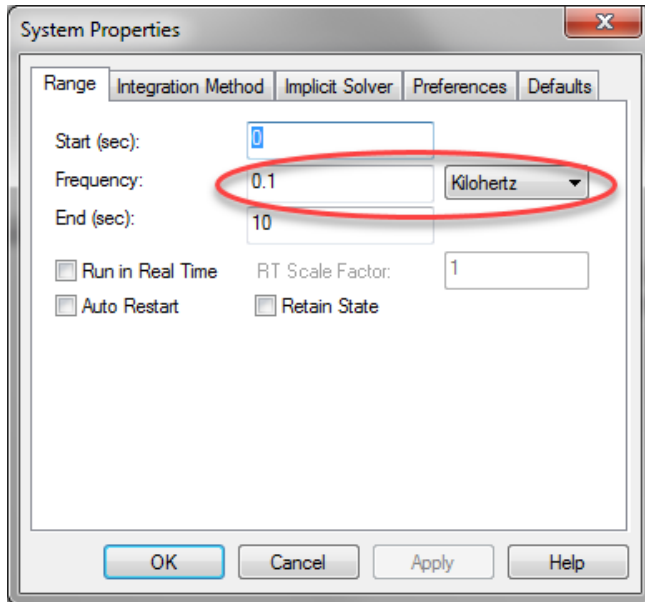
Note: The terms *tabs*, *pins*, and *connectors* are used interchangeably and refer to the input and output ports on the blocks. The tabs have different colors that indicate the type of data being passed. The input tab on the **Digital Output for Arduino** block is green, indicating integer values are accepted.

Setting simulation parameters

Before starting a simulation, you can set, among other things, the rate at which the diagram runs.

1. Choose **System > System Properties**.

The System Properties dialog is displayed.



2. In the **Frequency** box, enter **0.1** and select **Kiloherz**; then click **OK**.

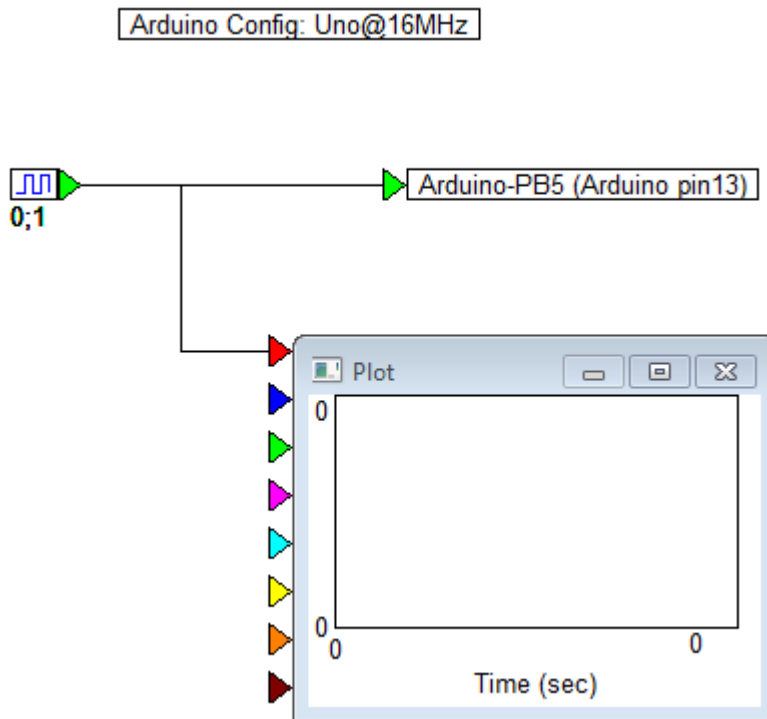
The diagram runs at 100 Hz.


Confirming the signal frequency

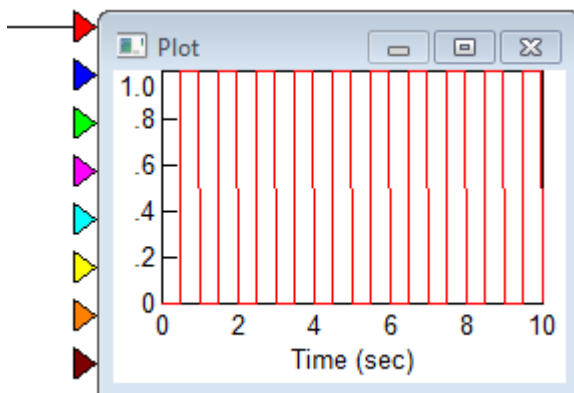
Although this diagram is very simple, it is good practice to wire a Signal Consumer block, like a plot block, into your diagram to check that the signals you are producing are what you expect.

1. Under **Blocks > Signal Consumers**, insert a **plot** block in the work area.
2. Connect the **squareWave** block to the **plot** block.

Your diagram will look like this:



3. Click  in the toolbar to start the simulation.



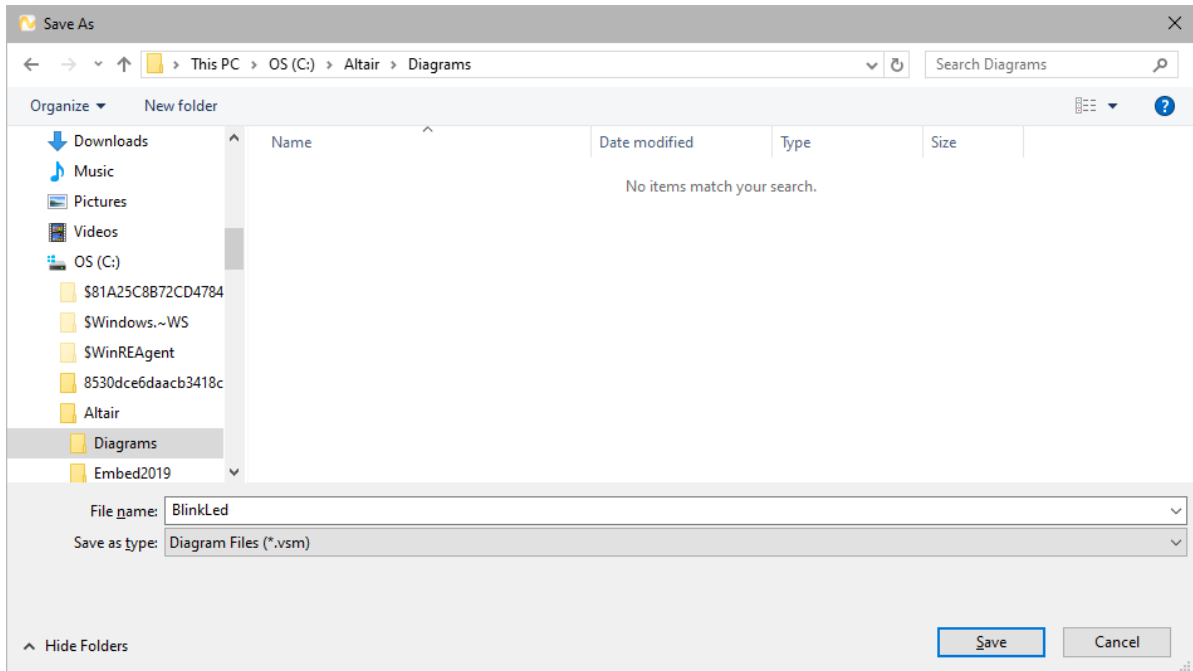
The plot trace shows that the signal correctly cycles between 0 and 1 at one second intervals.

Saving your work

At this point, it is a good idea to save your work using either **File > Save** or **File > Save As**. You can tell if your changes have not been saved if there is an asterisk after the diagram name in the title bar.

1. Click **File > Save As**.

The Save As dialog appears.



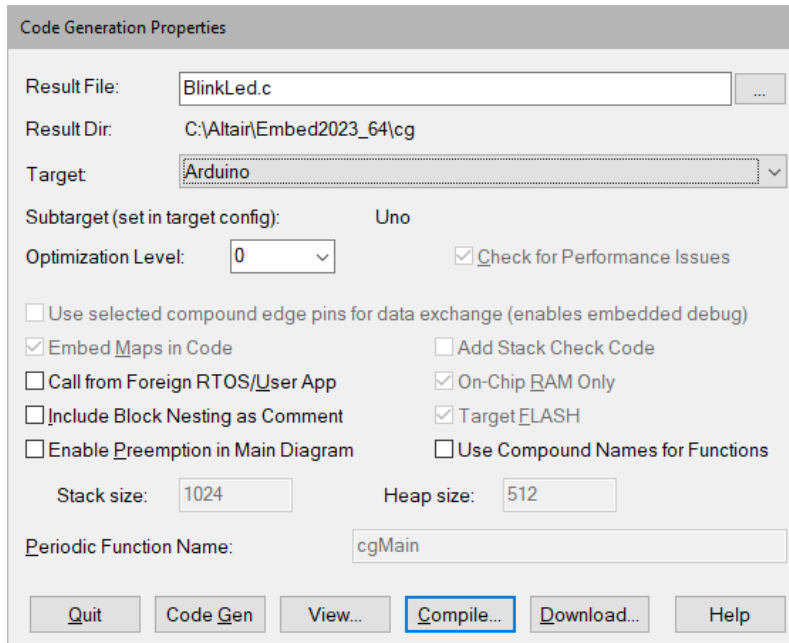
2. Navigate to where you want to save your diagram and enter *BlinkLed* in the **File name** text box; then click **Save**.

Compiling and linking your code

You are now ready to generate code to run on the Arduino Uno.

1. Click **Tools > Code Gen.**

The Code Gen dialog appears.



This dialog provides, among other things, the following information:

- **Result File:** The name of the generated C file. By default, Embed Personal uses the name of your diagram.
- **Result Dir:** The name of the directory in which the C file will be placed. C:\Altair\Embed2023\cg is the default directory.
- **Target:** The target device.
- **Subtarget:** The CPU that you selected when you [configured the diagram](#).

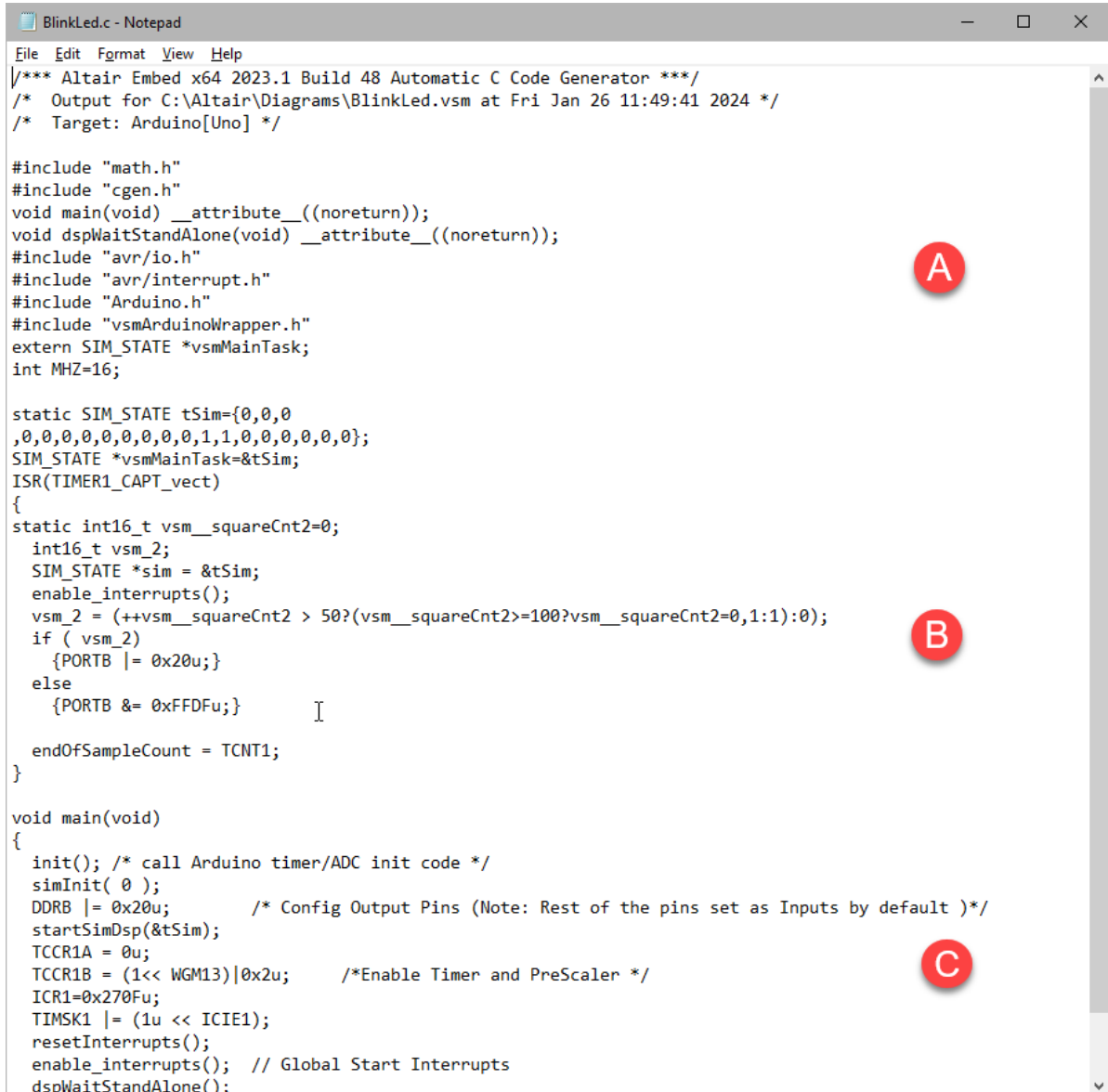
For this example, you can ignore the other parameters in the dialog.

2. Click **Compile**.

The following occurs:

- A BlinkLed.C file is generated.
- The target compiler generates a BlinkLed.ELF file (the target executable).

3. To examine the BlinkLed.C file, click **View** in the Code Gen dialog.



```
File Edit Format View Help
/**** Altair Embed x64 2023.1 Build 48 Automatic C Code Generator ****/
/* Output for C:\Altair\Diagrams\BlinkLed.vsm at Fri Jan 26 11:49:41 2024 */
/* Target: Arduino[Uno] */

#include "math.h"
#include "cgen.h"
void main(void) __attribute__((noreturn));
void dspWaitStandAlone(void) __attribute__((noreturn));
#include "avr/io.h"
#include "avr/interrupt.h"
#include "Arduino.h"
#include "vsmArduinoWrapper.h"
extern SIM_STATE *vsmMainTask;
int MHZ=16;

static SIM_STATE tSim={0,0,0
,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0};
SIM_STATE *vsmMainTask=&tSim;
ISR(TIMER1_CAPT_vect)
{
static int16_t vsm_squareCnt2=0;
int16_t vsm_2;
SIM_STATE *sim = &tSim;
enable_interrupts();
vsm_2 = (++vsm_squareCnt2 > 50?(vsm_squareCnt2>=100?vsm_squareCnt2=0,1:1):0);
if ( vsm_2)
{PORTB |= 0x20u;}
else
{PORTB &= 0xFFDFu;}
endOfSampleCount = TCNT1;
}

void main(void)
{
init(); /* call Arduino timer/ADC init code */
simInit( 0 );
DDRB |= 0x20u; /* Config Output Pins (Note: Rest of the pins set as Inputs by default)*/
startSimDsp(&tSim);
TCCR1A = 0u;
TCCR1B = (1<< WGM13)|0x2u; /*Enable Timer and PreScaler */
ICR1=0x270Fu;
TIMSK1 |= (1u << ICIE1);
resetInterrupts();
enable_interrupts(); // Global Start Interrupts
dspWaitStandAlone();
}
```

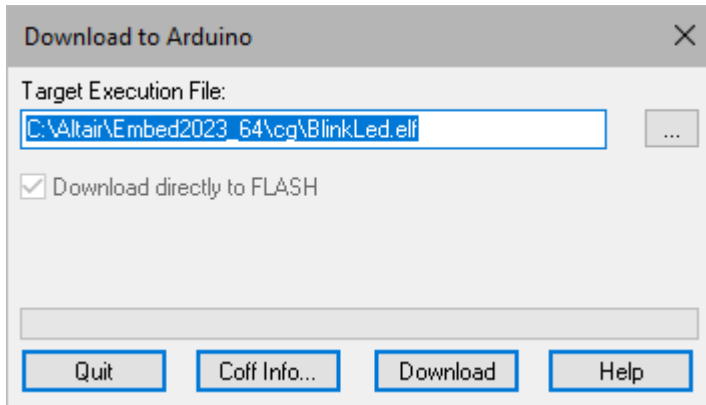
The code is separated into three sections:

- **A:** Includes the necessary header files for the code to run on the Arduino Uno.
- **B:** Sets the target interface to run at the rate specified in the System Properties dialog, which is 100 Hz, and creates 1 Hz blink (50 counts ON and 50 counts OFF).
- **C:** Generates interrupts at a 100 Hz rate.

Downloading and executing the code on the Arduino Uno

1. To download BlinkLed.ELF to the Arduino Uno, click **Download** in the Code Gen dialog.

The Download to Arduino dialog appears.

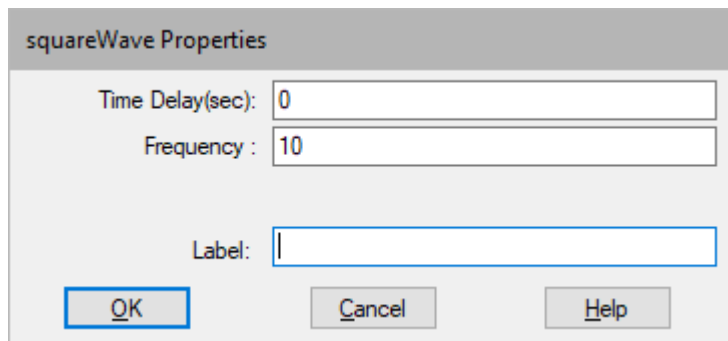


2. Click **Download**.

The built-in LED on the Arduino Uno starts blinking at one second intervals.

Changing the blink frequency

You can easily change the blink rate by right-clicking the squareWave block and editing the Frequency parameter.



In this case, the Frequency has been set to 10Hz. After you save the diagram, and [re-compile](#) and [download](#) the code to the Arduino, the built-in LED blinks at a more rapid rate.

Helpful resources

There are dozens of videos and sample diagrams that demonstrate basic and advanced operations you can perform on your hardware. There is also an online [Embed Forum](#) where you can ask questions, get answers, and promote conversations among other users and Embed developers and application engineers.

Videos

The [Altair YouTube Video center](#) includes a series of short videos that introduce key concepts that are useful when programming your hardware.

Sample diagrams

Sample diagrams are included in Embed Personal that supplement the online videos. To open and run these diagrams, go to **Examples > Embedded**, then click on the microprocessor you're interested in to see the corresponding sample diagrams.

Where to go from here

If you're interested in stepping up to Embed Pro for commercial projects, contact [Altair sales](#).